Onion Routing: Making a Traffic Analysis Resistant Network Robust and Fault-Tolerant

Michael George Reed
Department of Computer Science
University of Maryland

August 31, 1998

Abstract

Onion Routing is an infrastructure for private communication over a public network. It provides anonymous connections that are strongly resistant to both eavesdropping and traffic analysis. Onion routing's anonymous connections are bidirectional, near real-time, and can be used anywhere a TCP/IP socket connection can be used. This prospectus outlines the alpha onion routing system and defines a research path to overhaul the system for additional portability, extensibility, functionality, and fault-tolerance.

1 Introduction

Is Internet communication private? Most security concerns focus on preventing eavesdropping, i.e., outsiders listening in on electronic conversations. But encrypted messages can still be tracked, revealing who is talking to whom. This tracking and the inferences drawn are called traffic analysis and may reveal sensitive information. For example, the existence of inter-company collaboration may be confidential. Similarly, e-mail users may not wish to reveal who they are communicating with to the rest of the world. In certain cases anonymity may be desirable also: anonymous e-cash is not very anonymous if delivered with a return address [16]. Web based shopping or browsing of public databases should not require revealing one's identity.

A purpose of traffic analysis is to reveal who is talking to whom. The *anonymous connections* described here are designed to be resistant to traffic analysis, i.e., to make it difficult for observers to learn identifying information from the connection (e.g., by reading packet headers, tracking encrypted payloads, etc.). Any identifying information must be passed as data through the anonymous

connections. Onion routing provides bidirectional and near real-time communication similar to TCP/IP socket connections or ATM AAL5 [3]. The anonymous connections can substitute for sockets in a wide variety of unmodified Internet applications by means of proxies. Data may also be passed through a privacy filter before being sent over an anonymous connection. This removes identifying information from the data stream, to make communication anonymous too.

Although onion routing may be used for anonymous communication, it differs from anonymous remailers [20, 9] in two ways: Communication is real-time and bidirectional, and the anonymous connections are application independent. Onion routing's anonymous connections can support anonymous mail as well as other applications. For example, onion routing may be used for anonymous Web browsing. A user may wish to browse public Web sites without revealing his identity to those Web sites. That requires removing information that identifies him from his requests to Web servers and removing information from the connection itself that may identify him. Hence, anonymous Web browsing uses anonymized communication over anonymous connections. The Anonymizer [19] only anonymizes the data stream, not the connection itself. So it does not prevent traffic analysis attacks like tracking data as it moves through the network.

The remainder of the prospectus is organized in the following way: Section 2 presents an overview of onion routing; Section 3 examines prior work in the areas of traffic analysis, privacy, and anonymity; Section 4 defines the threat model for the system; Section 5 presents empirical data about the prototype system; Section 6 describes the research to be performed for the actual dissertation and the time-line to completion; Section 7 presents some concluding remarks.

2 Onion Routing Overview

In onion routing, instead of making socket connections directly to a responding machine, initiating applications make connections through a sequence of machines called *onion routers*. The *onion routing network* allows the connection between the *initiator* and *responder* to remain anonymous. Anonymous connections hide who is connected to whom, and for what purpose, from both outside eavesdroppers and compromised onion routers. If the initiator also wants to remain anonymous to the responder, then all identifying information must be removed from the data stream before being sent over the anonymous connection.

Onion routers in the network are connected by longstanding (permanent) socket connections. Anonymous connections through the network are multiplexed over the longstanding connections. For any anonymous connection, the sequence of onion routers in a route is strictly defined at connection setup. However, each onion router can only identify the previous and next hops along a route. Data passed along the anonymous connection appears different at each onion router, so data cannot be tracked en route, and compromised onion routers cannot cooperate by correlating the data stream each sees. We will also see that

they cannot make use of replayed onions or replayed data.

2.1 Operational Overview

The onion routing network is accessed via a series of *proxies*. An initiating application makes a socket connection to an *application proxy*. This proxy massages connection message format (and later data) to a generic form that can be passed through the onion routing network. It then connects to an *onion proxy*, which defines a route through the onion routing network by constructing a layered data structure called an *onion*. The onion is passed to the *entry funnel*, which occupies one of the longstanding connections to an onion router and multiplexes connections to the onion routing network at that onion router. That onion router will be the one for whom the outermost layer of the onion is intended. Each layer of the onion defines the next hop in a route. An onion router that receives an onion peels off its layer, identifies the next hop, and sends the embedded onion to that onion router. The last onion router forwards data to an *exit funnel*, whose job is to pass data between the onion routing network and the responder.

In addition to carrying next hop information, each onion layer contains an expiration time, selected ciphers, flags, and key seed material from which keys are generated for crypting[1] data sent forward or backward along the anonymous connection. (Define *forward* to be the direction in which the onion travels and *backward* as the opposite direction.)

Once the anonymous connection is established, it can carry data. Before sending data over an anonymous connection, the onion proxy adds a layer of encryption for each onion router in the route. As data moves through the anonymous connection, each onion router removes one layer of encryption, so it arrives at the responder as plaintext. This layering occurs in the reverse order for data moving back to the initiator. So data that has passed backward through the anonymous connection must be repeatedly post-crypted to obtain the plaintext.

By layering cryptographic operations in this way, the system gains an advantage over link encryption. As data moves through the network it appears different to each onion router. Therefore, an anonymous connection is as strong as its strongest link, and even one honest node is enough to maintain the privacy of the route. In link encrypted systems, compromised nodes can cooperate to uncover route information.

Onion routers keep track of received onions until they expire. Replayed or expired onions are not forwarded, so they cannot be used to uncover route information, either by outsiders or compromised onion routers. Note that clock skew between onion routers can only cause an onion router to reject a fresh

[1] Define the verb *crypt* to mean the application of a cryptographic operation, be it encryption or decryption.

onion or to keep track of processed onions longer than necessary. Also, since data is encrypted using stream ciphers, replayed data will look different each time it passes through a properly operating onion router.

Although the system is called onion routing, the routing that occurs here does so at the application layer of the protocol stack and not at the IP layer. More specifically, the system relies upon IP routing to route data passed through the longstanding socket connections. An anonymous connection is comprised of portions of several linked longstanding multiplexed socket connections. Therefore, although the series of onion routers in an anonymous connection is fixed for the lifetime of that anonymous connection, the route that data actually travels between individual onion routers is determined by the underlying IP network. Thus, onion routing may be compared to loose source routing.

Onion routing depends upon connection based services that deliver data uncorrupted and in-order. This simplifies the specification of the system. TCP socket connections, which are layered on top of a connectionless service like IP, provide these guarantees. Similarly, onion routing could easily be layered on top of other connection based services, like ATM AAL5.

2.2 Configurations

As mentioned above neighboring onion routers are neighbors by virtue of having longstanding socket connections between them, and the network as a whole is accessed from the outside through a series of proxies. By adjusting where those proxies reside it is possible to vary which elements of the system are trusted by users and in what way. (For some configurations it may be efficient to combine proxies that reside in the same place, thus they may be only conceptually distinct.)

2.2.1 Firewall Configuration

In the *firewall configuration*, an onion router sits on the firewall of a sensitive site. This onion router serves as an interface between machines behind the firewall and the external network. Connections from machines behind the firewall to the onion router are protected by other means (e.g., physical security). To complicate tracking of traffic originating or terminating within the sensitive site, this onion router should also route data between other onion routers. This configuration might represent the system interface from a typical corporate or government site. Here the application proxies (together with any privacy filters), and the onion proxies would typically live at the firewall as well. There are three important features of this basic configuration:

- Connections between machines behind onion routers are protected against both eavesdropping and traffic analysis. Since the data stream never appears in the clear on the public network, this data may carry identifying information, but communication is still private.

- The application proxies at the originating protected site knows both the source and destination of a connection. This protects the anonymity of connections from observers outside the firewall but also simplifies enforcement of and monitoring for compliance with corporate or governmental usage policy.

- The use of anonymous connections between two sensitive sites that both control onion routers effectively hides their communication from outsiders. However, if the responder is not in a sensitive site (e.g., the responder is some arbitrary Web server) the data stream from the sensitive initiator must also be anonymized. If the connection between the exit funnel and the responding server is unencrypted, the data stream might otherwise identify the initiator. For example, an attacker could simply listen in on the connections to a Web server and identify initiators of any connection to it.

2.2.2 Remote Proxy Configuration

What happens if an initiator does not control an onion router? If the initiator can make encrypted connections to some remote onion router, then he can function as if he is in the firewall configuration just described, except that both observers and the network can tell when he makes connections to the onion router. However, if the initiator trusts the onion router to build onions, his association with the anonymous connection from that onion router to the responder is hidden from observers and the network. In a similar way, an encrypted connection from an exit funnel to a responder hides the association of the responder with the anonymous connection .

Therefore, if an initiator makes an anonymous connection to some responder, and layers end-to-end encryption over that anonymous connection, the initiator and responder can identify themselves to one another, yet hide their communication from the rest of the world. This allows the building of virtual private networks without protected sites.

Notice, however, that the initiator trusts the remote onion router to conceal that the initiator wants to communicate with the responder, and to build an anonymous connection through other onion routers. The next section describes how to shift some of this trust from the first onion router to the initiator.

2.2.3 The Customer–ISP Configuration

Suppose, for example, an Internet Services Provider (ISP) runs an in-funnel that accepts connections from onion proxies running on subscribers' machines. In this configuration, users generate onions specifying a path through the ISP to the destination. Although the ISP would know who initiates the connection, the ISP would not know with whom the customer is communicating, nor would it be able to see data content. So the customer need not trust the ISP to maintain

her privacy. Furthermore, the ISP becomes a *common carrier*, who carries data for its customers. This may relieve the ISP of responsibility both for whom users are communicating with and the content of those conversations.

The ISP may or may not be running an onion router as well. If he is running an onion router, then it is more difficult to identify connections that terminate with his customers; however, he is serving as a routing point for other traffic. On the other hand, if he simply runs a funnel to an onion router elsewhere, it will be possible to identify connections terminating with him, but his overall traffic load will be less. Which of these would be the case for a given ISP would probably depend on a variety of service, cost, and pricing considerations.

Note that in this configuration the entry funnel must have an established longstanding connection to an onion router just like any neighboring onion router. But, in most other cases, where the funnel resides on the same machine as the onion router, establishing an encrypted longstanding connection should not be necessary since the funnel can be directly incorporated into the onion router.

3 Related Work

Chaum [2] defines a layered object that routes data through intermediate nodes, called *mixes*. These intermediate nodes may reorder, delay, and pad traffic to complicate traffic analysis. In mixes, the assumption is that a single perfect mix adequately complicates traffic analysis, but a sequence of multiple mixes is typically used because real mixes are not ideal. Because of this, mix applications can use mixes in fixed order, and often do. Onion routers differ from mixes in at least two ways: onion routers are more limited in the extent to which they delay traffic at each node because of the real-time expectations that the applications demand of socket connections. Also, in a typical onion routing configuration, onion routers are also entry points to the onion routing network, and traffic entering or exiting at those nodes may not be visible. This makes it hard to track packets, because they may drop out of the network at any node, and new packets may be introduced at each node. While onion routing cannot delay traffic to the extent that mixes can, traffic between onion routers is multiplexed over a single channel and is link encrypted with a stream cipher. This makes it hard to parse the stream.

Anonymous remailers like Penet [22] strip headers from received mail and forward it to the intended recipient. They may also replace the sender's address with some alias, permitting replies. These sorts of remailers store sensitive state: the mapping between the alias and the true return address. Also, mail forwarded through a chain of remailers may be tracked because it appears the same to each remailer.

Mix based remailers like [20, 9] use mixes to provide anonymous e-mail services. Essentially, the mail message is carried in the innermost layer of the onion

data structure. Another onion type structure, used for a return address, can be contained in the message. This makes the return path self contained, and the remailer essentially stateless. Onion routing shares many structures with Babel [9] but it uses them to build (possibly long lived) application independent connections. This makes anonymous connections accessible to a wide variety of applications. For application to e-mail it has both advantages and disadvantages. Onion routing's service makes an anonymous connection directly to the recipient's sendmail daemon. A disadvantage is that, since the connection is made in real-time, there is less freedom in mixing, which therefore might not be done as well. An advantage is that the anonymous connection is separated from the application, so anonymous e-mail systems are considerably simplified because the application specific part does not have to move data through the network. Furthermore, because the onion routing network can carry many types of data, it has the potential to be more heavily utilized than a network that is devoted only to e-mail. Heavy utilization is the key to anonymity.

In [4], a structure similar to an onion is used to forward individual IP packets through a network. By maintaining tracking information at each router, ICMP error messages can be moved back along the hidden route. Essentially, a connection is built for each packet in a connectionless service. Although a followup paper [5] suggests that performance will be good, especially with hardware based public key cryptography, empirical tests suggest that both the cryptographic overhead of building onions and the tracking of onions against replay is not efficiently done on a packet-by-packet basis. However, it is easy to imagine an onion routing proxy that collects IP packets and forwards them over some anonymous connection. In this way, communication is anonymous at the IP layer, but connections need not be built for each IP packet. This anonymous IP communication may be more robust than the onion routing architecture: it could survive a broken anonymous connection, since IP does not expect reliable delivery. In fact, we can build virtual private networks in this way over onion routing.

In [10], mixes are used to provide untraceable communication in an ISDN network. In a phone system, each telephone line is assigned to a particular local switch (i.e., local exchange), and switches are interconnected by a (long distance) network. Anonymous calls in ISDN rely upon an anonymous connection between the caller and the long distance network. These connections are made anonymous by routing calls through a predefined series of mixes within each switch. The long distance endpoints of the connection are then mated to complete the call. (Notice that observers can tell which local switches are connected.) Also, since each phone line has a control circuit connection to the switch, the switch can broadcast messages to each line using these control circuits. So, within a switch a truly anonymous connection can be established: A phone line makes an anonymous connection to some mix. That mix broadcasts a token identifying itself and the connection. A recipient of that token can make another anonymous connection to the specified mix, which mates the two

connections to complete the call.

The goal of anonymous connections over the Internet differs from anonymous remailers and anonymous ISDN. The data is different, with real-time constraints more severe than mail, but somewhat looser than voice. Both HTTP and ISDN connections are bidirectional, but, unlike ISDN, HTTP connections are likely to be small requests followed by short bursts of returned data. Most importantly, the network topology of the Internet is more akin to the network topology of the long distance network between switches, where capacity is a shared resource. In anonymous ISDN, the mixes hide communication within the local switch, but connections between switches are not hidden. This implies that all calls between two businesses, each large enough to use an entire switch, reveal which businesses are communicating. In onion routing, mixing is dispersed throughout the Internet, which improves hiding.

Pipe-net [21] is a proposal similar to onion routing that has never (and one might contend cannot ever) been built. Pipe-net's threat model is more paranoid than onion routing's: it attempts to resist active attacks by global observers. For example, Pipe-net's connections carry constant traffic, are all created at network setup, can never be destroyed, and any disruptions or degradations to any connection must be instantly propagated to all connections throughout the entire network.

The Anonymizer is a Web proxy that filters the HTTP data stream to remove a user's identifying information, essentially as the onion router filtering HTTP proxy does. For example, the Anonymizer will "strip out all references to your e-mail address, computer type, and previous page visited before forwarding your request" [19]. This makes Web browsing private in the absence of any eavesdropping or traffic analysis. The Anonymizer is vulnerable in three ways: First, it must be trusted. Second, traffic between a browser and the Anonymizer is sent in the clear, so that traffic identifies the true destination of a query, and includes the identifying information that the Anonymizer would filter. Third, even if traffic between the browser and the Anonymizer were encrypted, passive external observers could mount the volume attack mentioned in section 4.1. The Anonymizer, however, is now readily available to everyone on the Web.

NetAngels [24] is similar to the Anonymizer, except that it builds personal profiles of its subscribers and targets advertisements to match the profile. However, the profile is not released to the advertiser and is deleted when a subscription is canceled. Subscribers must trust NetAngels, and connections to the service are subject to the same attacks as the Anonymizer.

LPWA [23, 6] (formerly known as Janus) is a "proxy server that generates consistent untraceable aliases for you that enable you to browse the Web, register at web sites and open accounts, and be 'recognized' upon returning to your accounts, all while still *preserving your privacy.*" Like the previous two, the LPWA proxy is at a server that is remote from the user application. It is thus subject to the same trust and vulnerability limitations.

It is possible, however, to shift trusted elements to the user's machine (or to a

machine on the boundary between his trusted LAN and the Internet). Shifting trust in this way can improve the security of other privacy services like the Anonymizer, NetAngels, and LPWA. Currently, those are centralized to provide an intermediary that masks the true source of a connection. If anonymous connections are used to hide the source address instead, the other functions of these services may run as a local proxy on the user's desktop. Security is improved because privacy filtering and other services are done on a trusted machine and because communication is resistant to traffic analysis. Also, there is no central point of failure.

Another approach to anonymous Web connections is Crowds [15]. Crowds is essentially a distributed and chained Anonymizer, with encrypted links between crowd members. Web traffic is forwarded to a crowd member, who flips a weighted coin and, depending on the result, forwards it either to some other crowd member or to the destination. This system gives probable deniability, it does not give you privacy since all crowd members can see all traffic traversing them.

4 Threat Model

4.1 Threats

This section outlines the threat model. It does not intend to quantify the cost of attacks, but to define possible attacks. Future work will quantify the threat. First some vocabulary. A session is the data carried over a single anonymous connection. Data is carried in fixed length cells. Since these cells are multiply encrypted and change as they move through an anonymous connection, tracking cells is equivalent to tracking markers that indicate when cells begin. In a marker attack, the attacker identifies the set of out-bound connections that some distinguished marker may have been forwarded upon. By intersecting these sets for a series of distinguished markers belonging to the same session, an attacker may determine, or at least narrow, the set of possible next hops. In a timing attack, the attacker records a timing signature for a session that correlates data rate over time. A session may have a very similar timing signature wherever it is measured over a route, so cooperating attackers may determine if they carry a particular session.

It is assumes that the network is subject to both passive and active attacks. Traffic may be monitored and modified by both external observers and internal network elements, including compromised onion routers. Attackers may cooperate and share information and inferences. Furthermore, it is assumed roving attackers that can monitor part, but not all, of the network at a time.

The goal is to prevent traffic analysis, not traffic confirmation. If an attacker wants to confirm that two endpoints often communicate, and he observes that they each connect to an anonymous connection at roughly the same time, more

often than is statistically expected, it is reasonable to infer that the endpoints are indeed communicating. Notice that this attack is infeasible if endpoints live in protected networks behind trusted onion routers on firewalls.

If the onion routing infrastructure is uniformly busy, then passive external attacks are ineffective. Specifically, neither the marker nor timing attacks are feasible, since external observers cannot assign markers to sessions. Active attacks are possible since reducing the load on the system makes the network easier to analyze (and makes the system not uniformly busy).

Passive internal attacks require at least two compromised onion routers. Since onion routers can assign markers to a session, both the marker and timing attacks are possible. Specifically, timing signatures can be broadcast, and other compromised onion routers can attempt to find connections with matching timing signatures.

Another attack that is only feasible as an internal attack is the volume attack. Compromised onion routers can keep track of the number of cells that have passed over any given anonymous connection. They can then simply broadcast totals to other compromised onion routers. Cell totals that are close to the same amount at the same time at different onion routers are likely to belong to the same anonymous connection.

Active internal attacks amplify these risks, since individual onion routers can selectively limit traffic on particular connections. An onion router could, for example, force a particular timing signature on a connection, and advertise that signature.

4.2 Implementation Vulnerabilities

An implementation of a secure design can be insecure. This section describes several implementation decisions that were made for security considerations.

Onions are packaged in a sequence of cells that must be processed together. This onion processing involves a public key decryption operation which is relatively expensive. Therefore, it is possible to imagine an implementation that clears outgoing queues while an onion is being processed, and then outputs the onion. Therefore, any period of inactivity on the out-bound queues is likely to be followed by a sequence of onion cells being output on a single queue. Such an implementation makes tracking easier and should be avoided.

After processing at each onion router, onions are padded at the end to compensate for the removed layer. Similarly, the length and payload of a DESTROY command must be new random content at each onion router; otherwise, compromised onion routers could track that payload.

In a multi-threaded implementation, there is a significant lure to rely upon apparent randomness in scheduling to reorder events. If reordering is important to the secure operation of the system, deliberate reordering is crucial, since low level system randomness may in fact be predictable.

There are two vulnerabilities for which good solutions do not currently exist. If part of the onion routing network is taken down, traffic analysis may be simplified. Also, if a longstanding connection between two onion routers is broken, it will result in many DESTROY messages, one for each anonymous connection that was routed through that longstanding connection. Therefore, a compromised onion router may infer from near simultaneous DESTROY messages that the associated anonymous connections had some common route. Delaying DESTROY messages hurts performance, since it is required that a DESTROY message propagate to the endpoints to take down the connection that is visible to the user. Carrying the DESTROY message through the anonymous connection and garbage collecting dormant anonymous connections later would be ideal, but control information cannot be efficiently insert into a raw data channel, especially considering the layered encryption.

The system as a whole is susceptible to both lulls and spikes in traffic through any portion of the network. To mitigate the traceability of these events by external observers, the core onion routers pad and rate-cap the longstanding connections. Likewise, end-to-end padding and rate-caps can be controlled by both the in-funnel and exit-funnel.

5 Empirical Data

Readers are invited to experiment with the prototype of onion routing network by using it to anonymously surf the Web. For instructions please see `http://www.onion-router.net/`.

One should be aware that accessing a remote onion router does not completely preserve anonymity, because the connection between a remote machine and the first onion router is not protected. If that connection were protected, one would be in the remote proxy configuration, but there would would still be no reason to trust the remote onion router. If one had a secured connection to an onion router one trusted, the prototype onion router could be used as one of several intermediate routers.

The data presented below is for a network running on a single machine. In the experimental onion routing network, five onion routers run on a single Sun Ultrasparc 2 2170. This machine has two 167 MHz processors, and 256MB of memory. Anonymous connections are routed through a random sequence of five onion routers. Connection setup time should be comparable to a more distributed topology provided the machine does not become CPU bound. Data latency, however, is more difficult to judge. Clearly, data will travel faster over socket connections between onion routers on the same machine than over socket connections between different machines. However, on a single machine the removal or addition of layers of encryption is not pipelined, so data latency will probably be worse.

Onion routing's overhead is mainly due to public key cryptography and

is incurred while setting up an anonymous connection. On the Ultrasparc 2 running a fast implementation of RSA [1], a single public key decryption of a 1024 bit plaintext block using a 1024 bit private key and a 1024 bit modulus takes 90 milliseconds. Encryption is much faster, because the public keys are only 16 bits long. (This is why RSA signature verification is cheaper than signing). So, the public key cryptographic overhead for routes spanning five onion routers is just under 0.5 seconds. This overhead can be further reduced, either with specialized hardware, or even simply on different hardware (a 200 MHz Pentium would be almost twice as fast).

In practice, the connection setup overhead does not appear to add intolerably to the overhead of typical web connections. There is no reason that the same anonymous connection could not be used to carry the traffic for several 'real' socket connections, either sequentially or multiplexed. In fact, the specification for HTTP 1.1 defines pipelined connections to amortize the cost of socket setup, and pipelined connections would also transparently amortize the increased cost of anonymous connection setup.

Prototypes onion routing networks have been running since July 1997. Over a five month test period in 1998, more than 1.1 million Web connections were completed through the NRL prototype network from more than six thousand IP addresses in twenty countries and in all six main top level domains. Recent load statistics place us at close to 14,000 connections per day.

6 Dissertation Work

For the sake of brevity, many of the details of onion routing have been omitted from this prospectus but can be found in [7, 8, 11, 12, 13, 14, 17, 18]. Although I can claim the original kernel of the onion routing concept as my own, the bulk of this work has been a collaborative effort between Dr. David Goldschlag (formerly of NRL, now working for Divx), Dr. Paul Syverson (NRL), and myself with support from ONR, DARPA, and NRL. In light of this, I have chosen two original sections to explore within the system by myself: the Onion Routing Database Engine and Onion Router Clustering.

6.1 Onion Routing Database Engine

The original alpha onion routing system was highly vulnerable to breaks in the network graph of core onion routers. As detailed in section 4.2, breaks in the network graph result in the propagation of DESTROY messages throughout the network. Additionally, a break in the graph must be reported to every onion proxy.

The onion proxy is the critical component trusted with selecting a route through the network and building the onion which encapsulates that route. In order for to accomplish this, the onion proxy must know the state of the entire

network, know which exit points from the network would be most appropriate for a particular connection, and know all of the core onion router public certificates. It is reasonable to expect that the onion proxy may not know the entire graph at one time, or that the graph may be in a state of change therefore invalidating some of its knowledge. This may result in bad route selection which would require testing the onion and then selecting a new route.

Therefore, the network graph can be characterized as a dynamically changing, decentralized in nature of update origin, must be semi-consistent (i.e., we should resolve to a consistent state at all nodes, but some delay in propagation is allowed in which case different nodes will see a different picture of the world), capable of withstanding both outsider and insider attacks, and be distributed in an efficient manner. These constraints have lead to a highly distributed design utilizing a flood-fill (link state) algorithm to distribute the three types of information: public certificates, exit policies, and network connectivity.

A comprehensive analysis of the system still needs to be performed to determine whether or not the flood-fill algorithm fulfills the needs described above with emphasis on both security and performance.

6.2 Onion Router Clustering

Onion routing is highly dependent on public key cryptography, specifically RSA. Unfortunately, public key cryptography is extremely costly to perform computationally, and while hardware assist devices are becoming available, they tend to be exorbitantly expensive and lack the speed necessary to make a wide-spread deployment feasible. Another limitation shown above in section 6.1 is that changes in the network graph must be propagated to all onion proxies. Onion router clustering gracefully solves both of these problems while also giving us fault-tolerance in the design as a side benefit.

Clustering works by establishing a set of physically and/or logically separate (from a physical network prospective) core onion routers with the same identity (i.e. public certificate) and fully interconnecting all nodes within the set with the set's neighbors. When a neighbor to the cluster receives a new onion destined for the cluster, it flips a coin and forwards the onion to a particular cluster member. The cluster member process the onion as before.

A couple points are worth noting and must be explored:

- As long as one node within the cluster remains connected to its neighbors, the network graph need not change if individual nodes or links fail since from a macro prospective, the cluster is still connected. This reduces network load for updates to the graph and gains fault-tolerance from failure of nodes and links.

- Graceful increases and decreases in capacity can be accomplished. No longer will one need to run the latest and greatest hardware to get a performance boost.

- Detection of onion replay is no longer trivial since a malicious neighbors can choose which member of the cluster to forward an onion to. A new replay detection scheme must be developed for the clustering approach.
- Multiple parties are now in control of the same private certificate. From a security prospective, this is a bad policy to pursue. Countermeasures for detecting private certificate compromise will need to be explored.
- Because malicious neighbors are given a choice as to which cluster member to forward an onion to, the threat analysis will need to be extended to consider signaling information via that choice to a colluding neighbor on the other side of a cluster.

6.3 Dissertation Outline and Time-line

A breakdown of the proposed dissertation follows:

Chapter 1: Introduction and description of the problem
Chapter 2: Prior work
Chapter 3: Theoretical design of Onion Routing (alpha architecture)
Chapter 4: Engineering issues of Onion Routing (beta architecture)
Chapter 5: Threat Analysis of the beta architecture
Chapter 6: Database system design and extended threat analysis
Chapter 7: Clustering design and extended threat analysis
Chapter 8: Performance/usage/policy discussion
Chapter 9: Conclusions and future research directions

Chapter 1, 2, 3, and 4 would draw heavily on articles that I have co-authored and my internal notes/documentation. Chapter 5 would be based on a paper I am co-authoring with people inside NRL and elsewhere. Chapters 6 and 7 would be solely my contributions to the project and would be considered the novel portions of the work. Chapter 8 would most likely be my exclusive work, but that will be more reporting of findings and policy discussions rather than what I would term as security research.

The beta system is due to be delivered to ONR by the close of fiscal year '98. Much of the performance / usage / policy work will be conducted in fiscal year '99 under NRL funding. The comprehensive threat analysis paper is underway and should appear during calendar year '99. Most of the onion routing database engine research is complete, but the extended threat analysis will have to wait for the completion of the threat analysis paper. Clustering work will be performed during fiscal year '99. I expect all research and preliminary analysis to be complete by the close of calendar year '99 and to complete the dissertation in time to graduate during the spring of '00.

7 Conclusion

Anonymous connections are resistant to both eavesdropping and traffic analysis. They separate the anonymity of the connection from the anonymity of communication over that connection. Anonymous connections may be used as a new primitive that enables novel applications in addition to facilitating secure versions of existing services [12]. The onion routing network supporting anonymous connections can be configured in several ways, including a firewall configuration and a customer-ISP configuration, which moves privacy to the user's computer and may relieve the carrier of responsibility for the user's connections. Onion routing moves the anonymous communications infrastructure below the application level, properly separating communication and applications. Since the efficacy of mixes depends upon sufficient network traffic, allowing different applications to share the same communications infrastructure increases the ability of the network to resist traffic analysis.

References

[1] T. Acar, B. Kaliski, Jr., and Ç. Koç. "Analyzing and Comparing Montgomery Multiplication Algorithms", *IEEE Micro*, 16(3):26-33, June 1996.

[2] D. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", *Communications of the ACM*, v. 24, n. 2, Feb. 1981, pp. 84-88.

[3] D. Comer. *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture,* Prentice–Hall, Engelwood Cliffs, New Jersey, 1995.

[4] A. Fasbender, D. Kesdogan, O. Kubitz. "Variable and Scalable Security: Protection of Location Information in Mobile IP", 46^{th} *IEEE Vehicular Technology Society Conference*, Atlanta, March 1996.

[5] A. Fasbender, D. Kesdogan, O. Kubitz. "Analysis of Security and Privacy in Mobile IP", 4^{th} *International Conference on Telecommunication Systems Modeling and Analysis*, Nashville, March 1996.

[6] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. "How to Make Personalized Web Browsing Simple, Secure, and Anonymous", *Financial Cryptography '97*, February 1997, final proceedings to appear.

[7] D. Goldschlag, M. Reed, and P. Syverson. "Privacy on the Internet", INET '97, Kuala Lumpur, June 1997.

[8] D. Goldschlag, M. Reed, P. Syverson. "Hiding Routing Information", in *Information Hiding*, R. Anderson, ed., LNCS vol. 1174, Springer-Verlag, 1996, pp. 137–150.

[9] C. Gülcü and G. Tsudik. "Mixing Email with *Babel*", *1996 Symposium on Network and Distributed System Security*, San Diego, February 1996.

[10] A. Pfitzmann, B. Pfitzmann, and M. Waidner. "ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead", *GI/ITG Conference: Communication in Distributed Systems*, Mannheim Feb, 1991, Informatik-Fachberichte 267, Springer-Verlag, Heidelberg 1991, pp. 451-463.

[11] M. Reed, P. Syverson, and D. Goldschlag. "Proxies for Anonymous Routing", *Proc.* 12^{th} *Annual Computer Security Applications Conference*, San Diego, CA, IEEE CS Press, December, 1996, pp. 95–104.

[12] M. Reed, P. Syverson, and D. Goldschlag. "Protocols using Anonymous Connections: Mobile Applications", *Security Protocols, 5th International Workshop Proceedings*, B. Christianson, B. Crispo, M. Lomas, and M. Roe (editors), Springer-Verlag LLNCS 1361, 1998, pp. 13-23.

[13] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing," IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection, 1998.

[14] M. Reed, P. Syverson, and D. Goldschlag, "Onion Routing Network for Securely Moving Data through Communication Networks," patent pending, Navy Case 78,415.

[15] M. Reiter and A. Rubin. *Crowds: Anonymity for Web Transactions (preliminary announcement),* DIMACS Technical Reports 97-15, April 1997.

[16] D. Simon, "Anonymous Communication and Anonymous Cash", in *Advances in Cryptology–CRYPTO'96*, N. Koblitz, ed., LNCS vol. 1109, Springer-Verlag, 1996, pp. 61–73.

[17] P. Syverson, D. Goldschlag, and M. Reed. "Anonymous Connections and Onion Routing", *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, Oakland, CA, IEEE CS Press, May 1997, pp. 44–54.

[18] P. Syverson, M. Reed, and D. Goldschlag, "Private Web Browsing," Journal of Computer Security Special Issue on Web Security, Volume 5, Number 3, 1997, pp. 237-248.

[19] The Anonymizer. `http://www.anonymizer.com/`

[20] L. Cottrell. *Mixmaster and Remailer Attacks,* `http://obscura.obscura.com/~loki/remailer/remailer-essay.html`

[21] W. Dai. Pipe-net, February 1995, post to the `cypherpunks` mailing list.

[22] J. Helsingius. http://www.penet.fi/

[23] LPWA. http://www.lpwa.com/

[24] NetAngels. http://www.netangels.com/